HTTP MULTIPLEXOR/DEMULTIPLEXOR

Cross-Reference to Related Applications

This application is a continuation-in-part of U.S. Patent Application, Serial No. 09/882,375, entitled "HTTP Multiplexor/Demultiplexor," filed on June 15, 2001, which in turn claims priority from U.S. Provisional Patent Application, Serial No. 60/239,552, entitled "HTTP Multiplexor/Demultiplexor," filed on October 10, 2000. The disclosures of both these applications are incorporated herein by this reference, in their entirety and for all purposes.

Technical Field

The present invention relates generally to data transmission on computer networks, and more particularly to a networking device including a Hypertext Transfer Protocol (HTTP) Multiplexor/Demultiplexor.

Background of the Invention

The Internet has experienced explosive growth in recent years. The emergence of the World Wide Web has enabled millions of users around the world to easily download web resources containing text, graphics, video, and sound data while at home, work, or from remote locations via wireless devices. These web resources often are large in size and therefore require a long time to download, causing the user delay and frustration. Delay often causes users to abandon the requested web page and move on to another web page, resulting in lost revenue and exposure for many commercial web sites.

One cause of delay is accumulation of Hypertext Transfer Protocol (HTTP) requests within a Transfer Control Protocol (TCP) buffer of a server socket. When a user

requests a web page, a web browser sends HTTP requests to a server socket via an established TCP connection. When the server does not process requests to a socket quickly enough, HTTP requests build up in the TCP buffer for that socket, resulting in processing delay in that socket.

An additional cause of delay is socket-related overhead processing. Conventional networking systems open up a server socket for each client that connects to the server, so that server overhead tends to increase in proportion to the number of connected clients. A given server can efficiently handle only so much overhead at once. Accordingly, the one-socket-per-client approach fundamentally limits the number clients that can simultaneously access a server. This limitation is worse still in secure environments, which, due to key exchanges and other security-related considerations, involve even more overhead per socket.

Summary of the Invention

A system, method and device for multiplexing and demultiplexing HTTP requests and responses are provided. The method may include receiving HTTP requests from a plurality of clients and routing those requests to a single socket on a server system. The HTTP requests may be routed to a particular server socket based on socket response time, the type or size of data being requested, and/or on other parameters related to the HTTP requests. The method may also include receiving HTTP responses from the server system, and selectively routing those responses to corresponding clients.

According to another aspect of the invention, the method typically includes at an intermediate networking device, receiving HTTP requests from multiple originating

clients, multiplexing the HTTP requests, and sending the multiplexed HTTP requests to an optimal server socket. The method may further include receiving the HTTP responses from the server system, demultiplexing the HTTP responses, and sending the demultiplexed HTTP responses to corresponding originating clients.

The system typically includes a server system, plural clients configured to connect to the server system via a computer network, and a computer networking device positioned intermediate the server system and the clients on the computer network. The computer networking device typically has an HTTP multiplexor/demultiplexor configured to receive HTTP requests from more than one of the clients and to distribute those requests as a multiplexed transmission to a socket on the server system via a TCP connection.

The device typically includes an HTTP multiplexor/demultiplexor configured to receive HTTP requests from a plurality of clients and to distribute those requests in multiplexed form to a server system via a TCP connection. The device typically is further configured to receive HTTP responses from the server system, demultiplex the responses, and route the demultiplexed responses to corresponding clients.

Brief Description of the Drawings

Fig. 1 is a schematic view of a prior art network configuration.

Fig. 2 is a schematic view of a networking device and system according to the present invention, including an HTTP multiplexor/demultiplexor.

Fig. 3 is a schematic view of the networking device and system of Fig. 2, showing routing, multiplexing and demultiplexing features that the device and system may be configured to perform.

Fig. 4 is a schematic view of a client computing device that may be used with the device and system of Fig. 2.

Fig. 5 is a schematic view of one embodiment of the networking device of Fig. 2.

Fig. 6 is a schematic view of another embodiment of the networking device of Fig. 2.

Fig. 7 is a flowchart of a method of demultiplexing and multiplexing HTTP requests and responses according to one embodiment of the present invention.

Detailed Description of the Invention

Referring initially to Fig. 1, a prior art networking system is shown generally at 10. System 10 includes a plurality of remote clients 12 and a server 14. In system 10, a single Transport Control Protocol (TCP) connection 16 is established between each remote client 12 and server 14. Each TCP connection 16 is established between a single socket 16a on each remote client and a corresponding socket 16b on server 14, such that a one-to-one socket ratio is established. In other words, one server socket is opened for each connected client. Remote clients 12 send Hypertext Transfer Protocol (HTTP) requests via established TCP connections to a TCP buffer associated with server socket 16b. Requests received at the TCP buffer are processed only as quickly as the server can respond to them. Often, requests build up in the buffer because

the server cannot respond to them quickly enough, and server-side delay (also referred to as latency) often results. This is inefficient and frustrating, and may cause a user to abandon downloading the page.

In addition, each server socket 16b opened requires a certain amount of overhead processing by the server. This overhead processing tends to increase in proportion to the number of sockets that are opened, and thus in proportion to the number of clients connecting to the system. To maintain acceptable response times, the server system must limit the number of client computers that may be connected at any given time, typically by periodically timing out established client connections.

In Fig. 2, a system for processing HTTP requests according to one embodiment of the present invention is shown generally at 20. System 20 typically includes a plurality of remote clients 12 configured to connect with server system 22 via computer network 24 and networking device 26. As seen in Fig. 3, server system 22 may include one or more servers 14, including web servers, applications servers and the like. In a typically employed configuration, server system will host and provide access to a website 25. As will be explained in detail with reference to Fig. 3, connection between clients 12 and server system 22 is established via client TCP connections 120 on the "client side" of networking device 26 and server TCP connections 124 on the "server side" of the networking device. Networking device 26 may include an HTTP multiplexor/demultiplexor, as explained in more detail below, which is configured to route HTTP requests/responses and other network traffic between clients 12 and server system 22. This routing function may be configured to provide for combining or

multiplexing traffic received from multiple client connections onto a single connection to the server system. In these implementations, the system typically is also configured to separate, or demultiplex the traffic received along that single server connection, for selective routing and delivery to the appropriate connected clients.

Multiplexing and demultiplexing reduces the amount of server sockets necessary to service a given number of connected clients. The resulting reduction in socket-related overhead processing improves system performance and/or frees resources to allow more clients to connect to the system. In addition, where multiple server sockets are available, the system may be configured to further optimize performance through performance-based selection of server sockets. For example, HTTP requests may be selectively routed to a particular server socket based on optimal response time, in order to ensure efficient processing of the requests. It should be appreciated that the term socket, as used in connection with the present invention, refers to a port, buffer, logical node, or object configured to receive data in HTTP and other formats from a remote device via a network connection, and is not limited to a "socket" as defined in the Unix operating system environment.

Referring now to Fig. 4, remote client 12 typically is a personal computer including a processor 30 coupled to a communications bus 32. A mass storage device 34, such as a hard drive, CD ROM drive, tape drive, etc., and a memory 36 are also linked to the communications bus 32. Memory 36 typically includes random access memory (RAM) 38, and read-only memory (ROM) 40. ROM 40 typically includes a basic input/output system (BIOS) 42, which is configured to start up and operate basic

functions of the remote client. Remote client 12 typically is configured to access computer network 24 via a network interface 46. Alternatively, remote client 12 may be a portable data assistant, web-enabled wireless device, mainframe computer, or other suitable computing device.

Remote client 12 typically is configured to run an operating system (OS) 48 to manage programs or applications. Operating system 48 is stored in mass storage device 34. Examples of suitable operating systems include UNIX, Windows, MacOS, VMS, and OS/2, although virtually any suitable operating system may be used. Remote client 12 includes a browser program 50 stored in mass storage device 34 configured to display requested web resources to a user of remote client 12. Exemplary browser programs 50 include the Netscape browser commercially available from Netscape Communications Corporation of Santa Clara, California and the Internet Explorer browser commercially available from Microsoft Corporation of Redmond, Washington.

Server 14 also typically is a computer similar to that shown in Fig. 4. Server 14 typically includes a server program configured to communicate with remote clients using the HTTP protocol. The server program typically is configured to receive HTTP requests, and, in response send HTTP responses to browser 50 on remote client 12 via computer network 24.

Networking device 26 may be connected to server system 22 and remote clients 12 in various ways. On the client side, device 26 typically is connected to remote clients 12 via a public-type network, such as wide area network (WAN) 24, as seen in Figs. 5 and 6, which may form part of the Internet. The server-side link between device

26 and server system 22 typically is a private network such as an intranet or local area network (LAN) 84.

Web site 25 typically includes a collection of web resources typically located at a web address called a URL (Uniform Resource Locator), or at another form of URI (Uniform Resource Identifier). The term "web resource" refers generally to a data resource that may be downloaded or accessed by a web browser. Web resources may include web pages, executable code, scripts, graphics, video, sounds, text, and/or other data. Web resources may be static (e.g. stored file) or dynamic (e.g. dynamically generated output). Web resources may be stored on and served by a single server 14 or a number of servers 14. For example, images may be stored on one server while code may be stored in another server, and alternatively, copies of images and code may be stored on multiple redundant servers.

As shown in Fig. 5, networking device 26 typically includes a controller 70 having a memory 72 and processor 74 linked by a bus 76. Also coupled to bus 76 is a mass storage device 78 including a multiplexor/demultiplexor 80, which may also be referred to as a "mux/demux." Networking device 26 also typically includes a network interface 82 coupled to bus 76. Network interface 82 is configured to enable networking device 26 to communicate with client 12 via WAN 24 and with server system 22 via LAN 84. An example of a suitable network interface is the Intel Ethernet Pro 100 network card, commercially available from Intel Corporation of Santa Clara, California.

In Fig. 6, another embodiment of a networking device according to the present invention is shown generally at 26′. Networking device 26′ typically includes an

integrated circuit board 90. The integrated circuit board contains a bus 92 connecting a network interface 94, memory 96, processor 98, Application Specific Integrated Circuit (ASIC) 100, and mass storage device 102. Network interface 94 is configured to enable networking device 26′ to communicate with remote client 12 via WAN 24 and with server system 22 via LAN 84. ASIC 100 typically contains a multiplexor/demultiplexor 104. ASIC 100, processor 98, and memory 96 form a controller 106 configured to process requests for web resources according to the methods described below. It will be appreciated that the embodiments of networking device 26, 26′ may be a stand-alone network appliance or may be integrated with a web server. Additionally, the mass storage device of networking device 26, 26′ is typically Flash memory, ROM, or other form of non-volatile memory, although it will be appreciated that a hard drive or other drive may also be used.

As indicated above, networking device 26 typically is connected to server system 22 via LAN 84. Because the server-side connection is a private-type connection, while the client connections are public-type connections (e.g., WAN 24), networking device 26 may be considered a server-side proxy server. A proxy server is a program or device that acts as an intermediary between a browser and a server. Networking device 26 acts as an intermediary by receiving HTTP requests from remote clients 12 and sending those requests to a socket on server system 22, and by receiving server-generated HTTP responses and sending those responses to the remote client that originated the requests.

The networking devices of the present invention typically are provided with a software or firmware multiplexor/demultiplexor configured to route network traffic between remote clients and a server system. Typically, this is done by multiplexing HTTP requests received at network device 26 via multiple client TCP connections 120, so that those requests may be forwarded from device 26 as a multiplexed transmission to server system 22. This multiplexed transmission normally occurs via a single server TCP connection 124, so that only one socket is opened on the server side. Server system 22 in turn generates HTTP responses corresponding to the various requests of the remote clients 12. These responses are provided along a single server TCP connection 124 to device 26, where they are demultiplexed so that they may be selectively routed and delivered to the appropriate originating client 12.

The above description contemplates routing of all traffic from multiple client TCP connections onto a single server-side TCP connection. Alternatively, as will be seen with reference to Fig. 3, requests and responses may be routed to and from more than one server socket. It is preferable, however, that the number of server sockets used be fewer than the number of client connections, in order to reduce the per-socket overhead occurring on the server side of the system. As indicated above, overhead processing can significantly affect overall performance. One of the advantages of the present system is a reduction of socket-related server overhead, which permits individual servers to handle more client connections.

Referring now to Fig. 3, each remote client 12 has an associated client TCP connection 120 established with networking device 26, typically via a

telecommunications network such as WAN 24 (Fig. 2). Each client connection 120 includes a client socket 120a associated with one of client computers 12 and a client-side device socket 120b (or, simply client-side socket 120b) associated with networking device 26. Remote clients 12 are typically configured to send HTTP requests/responses 122 via connections 120.

Server system 22 is coupled with networking device 26 via various server TCP connections 124. Similar to the client side, each server connection 124 includes a server-side device socket 124b (or, simply server-side socket 124b) associated with networking device 26 and a server socket 124a associated with server system 22. As indicated, server system 22 may include a plurality of servers 14 configured to perform various functions. Server connections 124 and their associated sockets may be in a one-to-one relationship with servers 14, or multiple connections 124 may be associated with a given individual server.

The previously described multiplexing, demultiplexing and routing capabilities are more clearly seen with reference to Fig. 3. For, example, with respect to the top three client computers on the right side of the figure, the figure depicts combining (e.g., by multiplexing) the traffic associated with those client computers (e.g., request/response streams R1, R2 and R3) into an individual server TCP connection 124. This allows multiple clients 12 to connect to server system 22 using only one server socket 124a. This provides a significant advantage over conventional client-server connection schemes, which would require three separate server sockets to provide connections for all three clients. By using only one server socket, instead of three,

socket-related overhead processing on the server side is reduced, allowing a greater number of connected clients to be serviced at one time.

As indicated, networking device 26 is configured to selectively route HTTP requests and responses between client and server sockets. For example, as indicated by HTTP request/response streams R1, R2 and R3, networking device 26 is configured to receive HTTP requests from one or more of client computers 12, and selectively route those requests via one of server connections 124 to an individual server socket 124a. Networking device is also configured to receive HTTP responses from server system 22, and route those responses back to the appropriate originating client 12.

Networking device 26 may be configured to multiplex requests from multiple clients, for example by taking HTTP requests from multiple clients and routing those requests to server system 22 via a single server connection 124. For example, Fig. 3 depicts multiplexing of HTTP requests from the topmost three clients onto a single server connection 124. The process is typically referred to as multiplexing because the traffic from a number of client-side connections 120 is routed to a smaller number of server-side connections 124, and also because plural client-side connections 120 may be routed to a single server-side connection 124.

Various methods may be used to combine the requests for transmission via a single server-side connection 124. Typically, a multiplexing state agent, shown at M1-M6, is assigned to each client-side socket 120b on the networking device. Each multiplexing state agent is configured to, for each request received from the client, route the request to an optimal server-side socket 124b on the networking device, for

transmission to a server 14 of server system 22. When a response to the request is received from the server on the server-side socket 124b, the multiplexing state agent is configured to route the request back to the client-side socket 120b for the requesting client. The multiplexing state agent is free to route subsequent requests from the same client to an optimal server-side socket 124b, whether that be a different server socket, or the same server socket as used for previous requests. While typically each multiplexing state agent is configured to route requests from only one client-side socket 120b to an optimal one of a plurality of server-side sockets 124b, it will be appreciated that alternatively a single multiplexing state agent may be configured to route requests from more than one client-side socket 120b to (1) an optimal one of a plurality of server-side sockets 124b, and/or (2) a single server-side socket 124b.

Networking device 26 may be further configured to demultiplex the response stream received from server-side sockets 124b in response to the client requests. Specifically, a series of responses received from a particular server socket is processed by the multiplexing state agents managing transactions with that socket, to unbundle the response stream into discrete responses or streams corresponding to an individual one of clients 12. Each multiplexing agent is configured to detect those responses on the server-side socket 124b that correspond to requests from the client with which the agent is associated, and route the responses back to the originating client, via the client-side socket 120b for the originating client. This process is referred to as demultiplexing because a series of responses from a single server-side connection 124 is broken up and routed over a plurality of client connections 120 to a plurality of clients 12.

As indicated, client-side connections 120 may correspond with server-side connections 124 in various ways. For example, the system may be operated so that all client connections are multiplexed to an individual server connection. Where multiplexing is employed, networking device 26 is configured to multiplex HTTP requests provided from two or more client connections (e.g., the connections corresponding to R1, R2 and R3) into a single server connection, such that only one server socket need be opened. Alternatively, multiple server connections may be employed, where each server connection corresponds either to an individual client connection, or is multiplexed so as to correspond to multiple client connections. In any event, it will normally be desirable that the server connections be fewer in number than the client connections, in order to achieve an optimal reduction of socket-related overhead processing on the server side of the system. Regardless of the particular multiplexing configuration, networking device 26 is further configured to demultiplex the responses generated by server system 22, and cause those responses to be selectively routed to the appropriate originating client 12.

Where multiple server connections are available, various optimization schemes may be employed to reduce delay and otherwise improve performance. In particular, networking device 26 may be configured to route HTTP requests to an optimal server socket, which typically is a least busy server socket. To determine the optimal server socket, multiplexor/demultiplexor 80 (Fig. 5) may be configured to detect the response time at each server socket 124a by monitoring corresponding server-side sockets 124b on the networking device. The server socket with the fastest response time

may be determined to be the least busy server socket. Alternatively, or additionally, various sockets may be monitored to determine which server socket has the fewest unfulfilled requests. In addition, routing may be effected based on which of the server sockets was the last to be accessed.

In addition to or instead of the above-described optimization techniques, networking device 26 may be configured to determine the type of request being made and/or the type of data being requested and use that information to effect optimal routing. For example, all image requests may be handled by a predetermined set of sockets on server system 22, while all HTML requests may be handled by a different predetermined set of sockets. In addition, certain sockets may be designated to handle specified protocols or protocol versions. For example, one set of sockets could be designated for all HTTP 1.0 requests, with another set being designated to handle HTTP 1.1 requests.

Regardless of the particular implementation, these optimization techniques increase the overall efficiency and response time of server system 22 by adjusting the flow of requests away from slow, congested server sockets and toward fast congestion-free server sockets. These optimization techniques may be employed in the described networking devices of the present invention, in addition to or instead of the routing, multiplexing and demultiplexing features discussed above.

Typically, the connections between networking device 26, server system 22 and clients 12 (e.g., connections 120 and 124) are persistent TCP connections. Persistent TCP connections are connections that remain open until explicitly commanded to close or until the server times-out the connection. Alternatively, a connection other than a

persistent TCP connection may be used. Effective use of persistent connections is a significant advantage of the present invention over prior systems. Often the persistence feature is not used in conventional networking systems, or is only partially used, because of the significant amount of per-connection overhead placed on the system. As discussed above, this overhead fundamentally limits the number of clients that can be connected at any one time. Thus, to provide access to a large number of potential connected clients, many existing systems periodically terminate client connections, to allow others access to the system. This effectively is a disabling of the persistence feature available in newer networking protocols. The failure to leverage persistence is particularly a drawback in secure environments, such as SSL, where setting up and tearing down TCP connections involves key exchanges and other overhead intensive tasks relating to security. By reducing the excessive overhead that necessitates periodic terminating of client connections, networking device is able to establish and maintain persistent connections with clients and servers.

It will be appreciated that the described networking devices and systems are extremely flexible, and may be configured in a nearly limitless number of ways to enhance the performance of client-server networking systems. Other network device implementations are described in co-pending U.S. Patent Applications Serial Nos. 09/680,675, 09/680,997, and 09/680,998, filed October 6, 2000, Nos. 60/239,552 and 60/239,071, filed October 10, 2000, No. 60/287,188, filed April 27, 2001, and No. 60/308,234 filed July 26, 2001, and No. 60/313,006 filed August 16, 2001, the disclosures of each of which are herein incorporated by reference, in their entirety and for

all purposes. The features of these devices may variously be implemented in connection with the networking devices and systems of the present invention.

Turning to Fig. 7, a method 140 may be practiced according to the present invention. The steps of method 140 are typically accomplished by networking device 26, in connection with remote clients 12 and server system 22 communicating via the described network connections. Alternatively, the method may be accomplished by dedicated software on server system 14, or by some other suitable software or hardware device. At 142, the method typically includes establishing persistent TCP connections between networking device 262 and one or more sockets on server system 14. At 144, the method typically includes establishing persistent TCP connections between networking device 26 and one or more remote clients 12. At 146, the method further includes listening for HTTP requests from originating remote clients 12 and/or for HTTP responses from various server sockets.

When HTTP requests are detected at networking device 26, method 140 continues, at 148, with receiving HTTP requests at one or more client-side sockets 120b. Method 100 further includes, at 150, multiplexing HTTP requests so that the multiplexed requests may be transmitted to the server system via a single TCP connection. At 152, the method further includes routing the requests to the server system, typically by sending the multiplexed requests to an optimal server socket.

Prior to step 152, method 140 may also include monitoring server sockets to determine an optimal server socket. The optimal server socket may be determined by identifying a server socket with a least-lengthy response time. Alternatively, the optimal

server socket may be determined by determining a last-accessed server socket, determining a server socket with the fewest number of unfulfilled requests, determining the type or size of data being requested or other parameters related to the HTTP requests, or by weighing all or some of these conditions. By determining an optimal server socket, the multiplexor/demultiplexor is able to improve performance by facilitating more efficient use of server resources.

When HTTP responses are detected at the multiplexor/demultiplexor at step 146, method 140 proceeds to 154, and includes receiving HTTP responses from the server system. The multiplexor/demultiplexor typically is configured to determine the destination of the HTTP responses. At 156, the method includes demultiplexing HTTP responses received from server system, in order to permit selective routing and delivery of certain of those responses to the appropriate originating client. At 158, method 140 includes sending the demultiplexed responses to the originating remote client.

When there is a new remote client or server detected at 160, the method includes returning to step 144 to establish a persistent TCP connection with the new remote client, or returning to step 142 to establish a persistent TCP connection with the new server, respectively. It will also be appreciated that networking device 26 may be configured to establish a plurality of TCP connections with a plurality of servers and a plurality of remote clients, and therefore may be configured to handle HTTP requests and HTTP responses from multiple servers and remote clients at once.

Similar to the devices and systems described above, the described method enhances the performance of client-server networking systems. This is accomplished

through multiplexing and demultiplexing of HTTP requests/responses, in order to reduce overhead processing that results in conventional systems from opening a server socket for each client computer connecting to the server. This also allows for fuller utilization of the persistent connection features that are now available in connection with HTTP and other protocols. Additionally, or alternatively, the method may include selection of optimal server sockets, in order to further enhance server sufficiency.

While the present invention has been particularly shown and described with reference to the foregoing preferred embodiments, those skilled in the art will understand that many variations may be made therein without departing from the spirit and scope of the invention as defined in the following claims. The description of the invention should be understood to include all novel and non-obvious combinations of elements described herein, and claims may be presented in this or a later application to any novel and non-obvious combination of these elements. Where the claims recite "a" or "a first" element or the equivalent thereof, such claims should be understood to include incorporation of one or more such elements, neither requiring nor excluding two or more such elements.